Multi-Attribute Decision Making with Weighted Description Logics

Erman Acar, Manuel Fink, Christian Meilicke, Camilo Thorne and Heiner Stuckenschmidt
Data and Web Science Group
University of Mannheim
{erman,manuel,christian,camilo,heiner}@informatik.uni-mannheim.de

Abstract

We introduce a decision-theoretic framework based on Description Logics (DLs), which can be used to encode and solve single stage multi-attribute decision problems. In particular, we consider the background knowledge as a DL knowledge base where each attribute is represented by a concept, weighted by a utility value which is asserted by the user. This yields a compact representation of preferences over attributes. Moreover, we represent choices as knowledge base individuals, and induce a ranking via the aggregation of attributes that they satisfy. We discuss the benefits of the approach from a decision theory point of view. Furthermore, we introduce an implementation of the framework as a Protégé plugin called uDecide. The plugin takes as input an ontology as background knowledge, and returns the choices consistent with the user's (the knowledge base) preferences. We describe a use case with data from DBpedia. We also provide empirical results for its performance in the size of the ontology using the reasoner Konclude.

1 Introduction

The study of preference representation languages and decision support systems is an ongoing research subject in artificial intelligence, gaining more popularity every day. Since the inception of multi-criteria decision making (MCDM) and utility theory [16], numerous approaches have been studied, including probabilistic, possibilistic, fuzzy and graphical models [9, 23, 15, 13] amongst others. One approach that has been gaining in interest over the last two decades is the use of logical languages [5, 6, 11, 14, 17, 26, 22, 21] to encode preferences and decision-theoretic problems.

In this paper, we introduce a theoretical framework to encode single stage (or non-sequential) decision making problems using a weighted extension of Description

Vol. 4 No. 7 2017 pp. 1973 – 1995
IFCoLog Journal of Logic and its Applications

Logics (DLs). While we presented a first sketch of the framework in [2], we now extend our formalism and give a full presentation in this paper. Furthermore, we introduce our Protégé plugin uDecide which is based on this formalism. Our framework combines ontology reasoning (see [4]) to rank the available choices with respect to a set of weighted attributes that have been specified by the user; the greater the weight, the more important the attribute. This yields a compact representation for a user's preference over attributes. In that manner, it is an multi-attribute account to decision making problems. The framework may serve as a decision support system from three main perspectives: (1) in complex domains in which an extensive amount of background knowledge is required and it is hard to see the logical implications of any choice in terms of implying outcomes; (2) in scenarios which the size of the set of possible choices is too large for a human decision maker to operate; (3) when (1) and (2) are combined.

The required weighted DL (the *DL decision base*), can be built over any specific DL language. This provides flexibility in the sense that one can use tractable fragments e.g., the DL_{Lite} family [7] or $\mathcal{EL}$ [3] if scalability is important, or more expressive fragments if the domain to be modelled requires such, e.g., data types, a feature desirable to express numeric domains, common in the literature of decision theory (see [20]).

A particular feature of our approach is that we make a distinction between choices (alternatives, desired items or objects, etc.) and outcomes (which can be seen as the result of an ontological approach to decision making). In particular, an outcome is a subset of attributes which are represented by sets of description logic classes, and choices are *named individuals*. We assume that the preferences of the user are elicited (partially or completely) in the form of attributes, so that the preference relation of the user rather than talking about any specific choices e.g., a, b, talks about any anonymous or generic choices satisfying (i.e., instantiating) a given subset of attributes or criteria.

The framework and its implementation can be understood as a generic out-of-the-box expert system that turns an ontology for a specific domain into a powerful decision support system for the domain described by that ontology. Such an expert system can be applied to domains where expert knowledge is required e.g., to select between different treatments depending on characteristics and preferences of a patient (medical domain), or a location for building a power-plant depending on the preferences of corporate management (energy industry domain). Our approach might also be used, for example, as a web-based decision support/recommender system for e-shopping. For the aforementioned perspective (1), designed to stress the importance of logic reasoning, we will give a small example in which the agent decisions between two cars with different specifications. For the aforementioned

perspective (2), we will present a use case to illustrate, using the implemented system, uDecide, how to to generate reading proposals from thousands of choices. This use case will also demonstrate that it is very easy to apply our approach to any kind of domain for which an ontological representation or knowledge base is already available or can play a key role to encode domain knowledge.

In the remainder of the paper, we present the basics of (classical) utility theory and of DLs in Section 2.1. We assume that the reader has a basic familiarity with DL and omit an extensive introduction, which can be found in [4]. Next, we introduce the theoretical foundations of our framework in Section 3 and discuss it over an example. In the example, we encode a car buying agents decision making problem and show how our approach decides between two alternative cars according to the patient's preferences. In Section 4, we first give a general description of our plugin. Then we present a use case which is based on an excerpt from DBpedia that deals with books and authors. This use case illustrates how to convert an ontology into an expert system by applying uDecide. Furthermore, we report on first runtime results using Konclude [24] as reasoning engine in the background. In Section 5, we discuss related works. Finally, we conclude and give a brief outline for future research in Section 6.

2 Preliminaries

2.1 Preferences and Utility

Preferences are of central importance in the study of decisions. In related formal sciences such as mathematical economics, social choice theory and artificial intelligence, preferences are usually modelled as a binary relation over the set of choices $\mathcal{C}$ i.e., $c_1 \succeq c_2$, is read c_1 *is at least as good as* c_2 (for the agent which/who has the preference relation $\succ$), where $c_1, c_2 \in \mathcal{C}$. Other commonly used synonym terms are *outcome*, *alternative* and *object*. Moreover, it is often assumed that $\succeq$ is complete and transitive. The preference relations associated with $\succeq$ are defined as follows: for any $c, c' \in \mathcal{C}$,

$$c_1 \succ c_2 \quad \text{iff} \quad c_1 \succeq c_2 \text{ and } c_2 \not\succeq c_1, \qquad \text{(Strict preference)}$$
$$c_1 \sim c_2 \quad \text{iff} \quad c_1 \succeq c_2 \text{ and } c_2 \succeq c_1, \qquad \text{(Indifference)}$$

where the former is read c_1 *is better than* c_2, and the latter is read *the agent is indifferent between* c_1 *and* c_2. In order to represent the preference relation compactly, one introduces a *utility function* u (see [12]), which is a function that maps a choice to a real number reflecting the degree of desire. Observe that there can be more

than one such function which represents a preference relation. For the classical results that guarantees the existence of such functions, we refer to the so-called representation theorems, [12]. Formally, given a choice $c \in \mathcal{C}$, a *utility function*, $u : \mathcal{C} \to \mathbb{R}$ represents $\succeq$ if

$$c_1 \succeq c_2 \quad \text{iff} \quad u(c_1) \geq u(c_2). \tag{Utility function}$$

The associated preferences $\succ$ (strict preference) and $\sim$ (equivalent preference or indifference) are defined analogously. For instance, if $u(\textit{low-price}) = 20$ and on the other hand $u(\textit{high-price}) = 5$ this would induce the preference *low-price* $\succ$ *high-price* as $5 < 20$. Usually, choices are formalised as *values* or *elements* of *attribute*(s). Here, the choices *low-price* and *high-price* which represent any item of interest (e.g. a book, car or treatment) can be thought of as values of a single attribute *price*, or equally in set notation $\textit{price} = \{\textit{low-price}, \ \textit{high-price}\}$.

Most of the time, our decisions depend on more than a single attribute; for instance, if we intend to buy a book, we are interested usually not only in its price, but also in its author, which genre it belongs to, and even whether or not it even does have a prize. Multiattribute utility theory is the extended variant of utility theory that deals with such decision problems [16]. We will denote the set of attributes by $\mathcal{X}$ and refer to a specific attribute by $X_i \in \mathcal{X}$ where $i \in \{1, \ldots, |\mathcal{X}|\}$. Then, the set of choices is lifted to the cartesian product of the set of attributes that we read as (possible) outcomes, denoted Ω i.e., $\mathcal{C} \subseteq X_1 \times \ldots \times X_n = \Omega$. We say, $u : \Omega \to \mathbb{R}$ is the (multiattribute) utility function which represents $\succeq$ iff $\forall (x_1, \ldots, x_n), (y_1, \ldots, y_n) \in \Omega$,

$$(x_1, \ldots, x_n) \succeq (y_1, \ldots, y_n) \quad \text{iff} \quad u(x_1, \ldots, x_n) \geq u(y_1, \ldots, y_n).$$

Since the size of the Ω is large i.e., $|2^{\mathcal{X}}|$, making the assumption that u is *additive*, significantly decreases computational complexity. An additive function satisfies the following.

$$u(x_1, \ldots, x_n) = u(x_1) + \cdots + u(x_n) \tag{Additivity}$$

where $(x_1, \ldots, x_n) \in \Omega$.

The basic principle in utility theory is that a rational agent should always try to maximise its utility, or *should take the choice with the highest utility*, which is formalised as follows:

$$\delta_{opt}(\mathcal{C}) := \arg\max_{c \in \mathcal{C}} u(c) \tag{Optimal choice}$$

where $\delta : \mathcal{P}(\mathcal{C}) \to (\mathcal{C})$ is a choice function with the condition that $\emptyset \subset \delta(\mathcal{C}) \subseteq \mathcal{C}$ (it should always pick an element). The choice function δ_{opt} returns the *maximal*

elements w.r.t. $\succeq$ and u. Note that there can be more than one (as preference orderings are partial and utilities non-unique). The class of decision making problems which we limit ourselves with, are *discrete choice* problems. That is $\mathcal{C}$ is a finite and discrete set. Examples of such discrete choice problems include choosing a medical treatment program with respect to a patient's criteria, or selecting a location for a nuclear power plant following environmental and financial criteria. This setting is orthogonal to continuous set of choices (possibly a vector of arbitrary numerical quantities) which represents a choice as a real-valued optimization problem.

2.2 Description Logics

We assume that the reader has some familiarity with DL. If that is not the case, we refer the interested reader to [4]. The framework that we are presenting is independent from the choice of a specific DL language. In what follows we recall briefly the basics of DL, to clarify the notation used later on defining the framework, the examples and, last but not least, our plugin.

DL signatures can be thought of as triples (N_C, N_R, N_I), where N_C is the set of atomic concepts, N_R is the set of role names, and N_I is the set of individuals. Along the text, we assume the *unique name assumption*, which means that different individuals have different names. We denote concepts by C and D, roles by r and S, and individuals as a and b. Concept descriptions are defined inductively from N_C as $\neg C$, $C \sqcap D$, and $C \sqcup D$ if C and D are concept descriptions, and $\exists r.C$ and $\forall r.C$ if $r \in N_R$ and C is a concept description. The top concept $\top$ is abbreviation for $C \sqcup \neg C$ and the bottom concept $\bot$ is for $\neg\top$. An interpretation is a pair $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where the domain $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is interpretation function that assigns to every concept name C a set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every role name R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. It is defined inductively for every concept description as follows; $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\exists r.C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} \mid \text{exists } b, (a,b) \in r^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}$, and $(\forall r.C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} \mid \text{for all } b, (a,b) \in r^{\mathcal{I}} \text{ implies } b \in C^{\mathcal{I}}\}$. Any other extension is defined accordingly, and will be clarified when it is necessary.

In DLs, there is a distinction between *terminological knowledge* (TBox) and *assertional knowledge* (Abox). TBox is a set of concept inclusion axioms: $C \sqsubseteq D$ where the interpretation is $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $C \equiv D$ if $C \sqsubseteq D$ and $C \sqsubseteq D$. ABox is a set of concept assertions $C(a)$ where $a \in N_I$ and $C(a)^{\mathcal{I}} := a^{\mathcal{I}} \in C^{\mathcal{I}}$, and role assertions $R(a,b)$ where $(a,b) \in N_I \times N_I$ and $R(a,b)^{\mathcal{I}} := (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

A concept is satisfiable if there is an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. A concept is satisfiable with respect to $\mathcal{T}$ if and only if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. A concept inclusion $C \sqsubseteq D$ is said to be satisfiable if and only if there

is an $\mathcal{I}$ which respects $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (i.e. $I \models C \sqsubseteq D$). A concept C is subsumed by a concept D with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model of $\mathcal{I}$ of $\mathcal{T}$ (i.e. $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$). If $\mathcal{T}$ is a set of axioms, then $\mathcal{I}$ is a model of $\mathcal{T}$ if and only if $\mathcal{I}$ satisfies every element of $\mathcal{T}$. Such a TBox is called *coherent*. We say that an assertion α is entailed by ABox $\mathcal{A}$ (i.e. $\mathcal{A} \models \alpha$) if every model of $\mathcal{A}$ also satisfies α. One basic reasoning service we will use is *instance check*; to check for a given ABox $\mathcal{A}$ and α, weather $\mathcal{A} \models \alpha$ holds. An ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$ if there is a model of both $\mathcal{T}$ and $\mathcal{A}$. We call the pair $\mathcal{K} := \langle \mathcal{T}, \mathcal{A} \rangle$ a knowledge base, and also say that K is satisfiable if $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$.

A concrete domain $\mathcal{D}$ is a pair $(\Delta^{\mathcal{D}}, pred(\mathcal{D}))$ where $\Delta^{\mathcal{D}}$ is the domain of $\mathcal{D}$ and $pred(\mathcal{D})$ is the set of predicate names of $\mathcal{D}$. It is assumed that $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{D}} = \emptyset$, and each $P \in pred(\mathcal{D})$ which is of arity n, is associated with $P^{\mathcal{D}} \subseteq (\Delta^{\mathcal{D}})^n$. We will denote *functional roles* with lower case r. In DL with concrete domains, it is assumed that N_R is partitioned into a set of *functional roles* and the set of ordinary roles. A role r is *functional* if for every $(x, y) \in r$ and $(w, z) \in r$ it implies that $x = w \implies y = z$. Functional roles, in the extended language, is interpreted as partial functions from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{D}}$. Functional roles and ordinary roles are both allowed to be used with both the existential quantification and the universal quantification. A concrete domain is required to be closed under negation (denoted by $\overline{P}$), in order to be able to compute the negation normal form of the concepts defined via extended constructs.

3 Weighted DLs for Decision Making

In this section, we introduce the theoretical underpinning of our plugin, which is a framework based on weighted DLs. In a loose sense, we will follow a specific-to-generic path while introducing definitions, ending up defining the generic framework, the DL *decision base*.

As an ontological approach to decision making, our aim is to use an *a priori* preference relation over attributes (ontological classes) to derive an *a posteriori* preference relation over choices (ontological individuals). To this end, we will define *a priori* (e.g., whose values are given by the user) a utility function U defined over $\mathcal{X}$. Then we will extend it to the subset of attributes, via a utility function u defined over choices using logical entailment. The use of lower case u, has two motivations: *i*) dealing with a technical subtlety, that is, a choice is an individual while a corresponding outcome is a set of classes (which are mathematically different types of objects), and *ii*) this gives flexibility to aggregate U in different forms e.g., *max*, *mean*, or a form which is a customized arbitrary utility function.

3.1 Modeling Attributes as DL Concepts

We represent each attribute in the original decision making problem by a class. Furthermore, for every value of an attribute in the original decision making problem, we introduce a new (sub)class to the set of classes at hand. For instance, if *colour* is an attribute in the decision making problem we would like to model, we simply represent it by the class $Colour$ (i.e., $Colour \in \mathcal{X}$), and if *blue* was a value of the attribute *colour* in the original decision making problem, we extend our attribute set $\mathcal{X}$ simply by adding the class $Blue$, which is a subclass of $Colour$. Note that this decomposition process will yield a binary term vector for $\mathcal{X}$. This is indeed our aim since in the sequel the aggregation of utilities of classes will be done with respect to the entailed class membership (i.e., $\mathcal{K} \models X(c)$) which has two possible cases in return : an individual c (as choice) is either a member of the class X or not.

We assume a total and transitive preference relation (i.e., $\succeq_{\mathcal{X}}$) over an ordered set of attributes $\mathcal{X}$ that are not necessarily atomic, and a function $U : \mathcal{X} \to \mathbb{R}$ that represents $\succeq$ (i.e., $U(X_1) \geq U(X_2)$ iff $X_1 \succeq_{\mathcal{X}} X_2$ for $X_1, X_2 \in \mathcal{X}$). The function U can be thought of as a weight function, which assigns an *a priori* weight to each class $X \in \mathcal{X}$, therefore makes the description logic *weighted*. We denote **the utility of a class** $X \in \mathcal{X}$ by $U(X)$. This reflects an agent's preference relation over the set of attributes $\mathcal{X}$. The greater the utility an attribute has, the more preferable the attribute is. Furthermore, we partition the attribute set $\mathcal{X}$ into two subsets; *desirable* that is the set of attributes with non-negative weights, denoted $\mathcal{X}^+$, and *undesirable* $\mathcal{X}^-$, i.e., $X \in \mathcal{X}$ iff $U(X) \geq 0$ and $\mathcal{X} = \mathcal{X}^+ \cup \mathcal{X}^-$ with $\mathcal{X}^+ \cap \mathcal{X}^- = \emptyset$. Intuitively, any attribute that is not desirable is undesirable, and a zero-weighted attribute can be interpreted as desirable with zero utility.

3.2 From Utility of Criteria to Utility of Choices

We call N_I as the set of named individuals. A *choice* is an individual $c \in N_I$. We denote by $\mathcal{C}$ the finite set of choices. In order to derive a preference relation (*a posteriori*) over $\mathcal{C}$ (i.e., $\succeq_{\mathcal{C}}$) which respects $\succeq_{\mathcal{X}}$, we will introduce a utility function $u(c) \in \mathbb{R}$, which measures **the utility of a choice** c relative to the attribute set $\mathcal{X}$ and the utility function U over attributes as an aggregator. For simplicity, we will abuse the notation and use the symbol $\succeq$ for both choices and sets of attributes whenever it is obvious from the context. In the following, we define a particular u, which we call σ-utility. It is intuitively defined relative to a knowledge base.

Definition 3.1 (σ-utility of a choice)**.** *Given a consistent knowledge base $\mathcal{K}$, and a set of choices $\mathcal{C}$,* utility of an choice $c \in \mathcal{C}$ *is*

$$u_\sigma(c) := \sum\{U(X) \mid X \in \mathcal{X} \text{ and } \mathcal{K} \models X(c)\}.$$

It is easy to see that u_σ induces a preference relation over $\mathcal{C}$ i.e., $u_\sigma(c_1) \geq u_\sigma(c_2)$ iff $c_1 \succeq c_2$. Also, notice that each choice corresponds to a set of attributes, that the membership is logically entailed e.g., $\mathcal{K} \models X(c)$. It can be easily be that such a summation function coincide with an additive multiattribute function given in previous section in the sense that every choice c corresponds to an outcome.

Following DL terminology and putting things together, we introduce the notion of a generic UBox, denoted $\mathcal{U}$ that is the component we need to generate utility functions.

Definition 3.2 (UBox)**.** *A UBox is a pair $\mathcal{U} := (u, U)$, where U is a utility function defined over $\mathcal{X}$ and u is a utility function defined over $\mathcal{C}$.*

Next, we introduce the key notion of *decision base*, which can be interpreted as a (formal, logical) model for an artificial agent in a decision situation, or a decision support system. A decision base is a quadruple which consists of a consistent *background knowledge*, a DL knowledge base $\mathcal{K}$, a finite set of available *choices* $\mathcal{C}$ which is represented as a set of individuals, a utility box, the component to encode user preferences and to generate a respective a utility function, and a choice function which defines the selection criterion from the set of choices $\mathcal{C}$. A generic decision base is defined formally as follows.

Definition 3.3 (Decision Base)**.** *A* decision base *is a quadruple $\mathcal{D} = (\mathcal{K}, \mathcal{C}, \mathcal{U}, \delta)$ where:*

- *$\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is a consistent knowledge base, $\mathcal{T}$ is an acyclic TBox and $\mathcal{A}$ is an ABox,*
- *$\mathcal{C} \subseteq N_I$ is the set of choices,*
- *$\mathcal{U} = (u, U)$ is a UBox,*
- *δ is a choice function.*

Informally, the role of $\mathcal{K}$ is to provide assertional information about the choices at hand, along with the general terminological knowledge information that the agent may require to reason further over choices. Note that this is just as in the case of σ-utility, that is meant to measure the value of a choice with respect to the classes (possibly deduced) which it belongs. The following proposition is an immediate result of that. For simplicity, we will drop the symbol σ and write u instead.

Proposition 1. *Let $\mathcal{K}$ be a knowledge base of $\mathcal{D}$ and $c_1, c_2 \in \mathcal{C}$ be any two choices.* ***i)*** *If for every $X_1 \in \mathcal{X}^+$ with $\mathcal{K} \models X_1(c_1)$, there is a $X_2 \in \mathcal{X}^+$ with $\mathcal{K} \models X_2(c_2)$ such that $\mathcal{K} \models X_1 \sqsubseteq X_2$, then $c_1 \succeq c_2$.* ***ii)*** *$\mathcal{X}^+$ is mutatis mutandis replaced by $\mathcal{X}^-$ then $c_2 \succeq c_1$.*

Proof. (*Sketch*) The proof proceeds by induction on X_2 (a DL concept). We omit the basis and deal only with one inductive case. **i)** Let $X_1 \in \mathcal{X}^+$ be any attribute such that $\mathcal{K} \models X_2(c_2)$. Assume that X_2 is of the form $X_2 \sqcap X_2'$. Since $\mathcal{K} \models X_1 \sqsubseteq X_2 \sqcap X_2'$ by assumption, it follows that $\mathcal{K} \models X_2 \sqsubseteq X_2$ and $\mathcal{K} \models X_1 \sqsubseteq X_2'$. Hence, by IH on either X_2 or X_2', $c_1 \succeq c_2$. **ii)** Similar to the previous case. □

Moreover, in this work, we will do not bother ourselves with the general details of the choice function, since we will consider only one, that is the *maximality criterion* (picking up the choice(s) with the maximum utility) which is defined modularly for $\mathcal{D}$ just as aforementioned δ_{opt} in previous section.

The given ontological approach to decision making provides an intuitive relation between attributes. Assume that we limit ourselves to desirable attributes. Then *ceteris paribus* any thing that belongs to a class should be at least as desirable as something that belongs to a superclass. For instance, a *new sport car* is at least as desirable as a *sport car* (since anything that is a *new sport car* is a also *sport car* i.e., *new sport car* $\sqsubseteq$ *sport car*). Note that the opposite would be the case if the concerned attributes were undesirable. The following corollary is a natural result of this approach to decision theory, which says that two choices are of same desirability (i.e., indistinguishable w.r.t. desirability) if they belong to exactly the same classes.

Corollary 1 (Indistinguishableness)**.** *Let $\mathcal{D}$ be a decision base, then for any $c_1, c_2 \in \mathcal{C}$, $c_1 \sim c_2$ iff $\{X_1 \in \mathcal{X} \mid \mathcal{K} \models X_1(c_1)\} = \{X_2 \in \mathcal{X} \mid \mathcal{K} \models X_2(c_2)\}$.*

Proof. By applying Proposition 1 in both directions. □

The intuitive explanation for Corollary 1 is that we measure the desirability (and non-desirability) of things, according to what they are, or which classes they do belong to. This brings forward the importance of reasoning, since it might not be obvious at all that two choices actually belong to exactly the same classes as attributes.

Decision bases offer high flexibility in representing preferences, due to having both qualitative (logical) and quantitative (weights) components. The following example illustrates their main properties.

3.3 Example: Car Buyer

Consider an agent who wants to buy a second hand sports car. After visiting various car dealers, he finds two alternatives as fair deals; a sport Mazda (*Mx-5 Miata Roadster, 2013*) which fits his original purpose and a BMW (*335i Sedan, 2008*) which is also worth considering since it has a very strong engine (*300 horsepower (hp)*) and also comes with a sport kit. The car buyer's decision base (background

knowledge $(\mathcal{T}, \mathcal{A})$, choices $\mathcal{C} = \{car_1, car_2\}$, and attributes mentioned in $\mathcal{U}$) is as in Figure 1.

As the use of numerical domains is common to classical Decision Theory, we will use the language with concrete domains. If the reader is already familiar with concrete domains, she can skip the technical definitions and go directly to Figure 1.

Let us clarify concrete domains and predicates which are used in the example. We take the concrete domain Car and $\Delta^{Car} := \Delta^{\$} \cup \Delta^{sec} \cup \Delta^{mpg} \cup \Delta^{mph} \cup \Delta^{hp}$ with $\Delta^{\$} \cap \Delta^{sec} \cap \Delta^{mpg} \cap \Delta^{mph} \cap \Delta^{hp} = \emptyset$, and $pred(Car) := pred(\$) \cup pred(mpg) \cup pred(mph) \cup pred(sec)$. We define the partition (of the domain Δ^{Car}) $\Delta^{\$}$ as a countably infinite set $\{i\$ \mid i \in \mathbb{N}\}$, $pred(\$) := \{<_{\$}, >_{\$}, \geq_{\$}, \leq_{\$}, :=_{\$}, \neq_{\$}\}$. $(<_{\$})^{\$}(x, y) = \{(x, y) \in \Delta^{\$} \times \Delta^{\$} \mid i, j \in \mathbb{N}$ with $x := i\$$ and $y := j\$$ such that $i < j\}$. Other predicates are defined similarly in an obvious way parallel to usual binary relations over $\mathbb{N}$. For convenience, we extend $pred(\$)$ with finitely many unary predicates in the form of $<_x := \{\forall y \in \Delta^{\$} \mid <_{\$} (x, y)\}$ and also of $>_x$, $\leq_x$, $\geq_x$, $=_x$, $\neq_x$ which are similarly defined, enough to express the intended TBox. Note that $pred(\$)$ is closed under negation: $\overline{<_{\$}}(x, y) = \geq_{\$} (x, y)$, etc. For other partitions, we take $\Delta^{sec} := \{i\ sec \mid i \in \mathbb{R}^+ \setminus \{0\}\}$, $\Delta^{mpg} := \{i\ mpg \mid i \in \mathbb{N}\}$, $\Delta^{mph} := \{i\ mph \mid i \in \mathbb{N}\}$, $\Delta^{hp} := \{i\ hp \mid i \in \mathbb{N} \setminus \{0\}\}$. The rest of the respective predicate names and functional roles are defined in an obvious way ($hasPrice^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\$}$, $hasKit : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, etc).

According to the agent, taking $\mathcal{T}$ into account, a *Bmw* is a *prestigious car*. Considering a *200 hp* or above is enough to refer to a car as strong. An economic car should go for more than *20 miles per gallon* (*mpg*). A car is *new* if it was manufactured in *2012* or later.

Considering U in Figure 1, the agent (car buyer) is more interested in having a *prestigious car* than having an *inexpensive car*. He prefers *convertible* to *sedan*. However, these are not as important as a car to be an *economic car*, or a *strong car*. Using the given decision base, we can calculate the utility of each choice ($u_\sigma(car_1) = 220$, $u_\sigma(car_2) = 170$), which implies (by the assumption: the higher the utility, the more desirable is the choice) that $car_1 \succ car_2$.

4 System Description

We implemented our approach as Protégé plugin available at the link `https://code.google.com/p/udecide/`. We first briefly describe the functionality and architecture of our plugin in Section 4.1. Then we present a use case that shows how a user interacts with the plugin in Section 4.2. This use case also illustrates how the plugin can be used as an out-of-the-box expert system for any knowledge domain available as ontology.

$\mathcal{T} = \{\exists hasPrice.\leq_{30000\ \$} \equiv InexpensiveCar,$
$ExpensiveCar \sqsubseteq HighClassCar,$
$HighClassCar \sqsubseteq PrestigiousCar,$
$\exists hasFuelConsumpt.\geq_{20mpg} \equiv EconomicCar,$
$Roadster \sqsubseteq PrestigiousCar,$
$MiddleClassCar \sqcap HighClassCar \sqsubseteq \bot,$
$SportsCar \sqcup \exists hasHP.\geq_{200hp} \sqsubseteq StrongCar,$
$2Doors \sqcap 4Doors \sqsubseteq \bot,$
$\exists has0-60mph.\leq_{7.0sec} \sqcap$
$\exists hasHP.\geq_{270hp} \sqsubseteq VeryStrongCar,$
$2Doors \sqcap \neg Convertible \equiv Coupé,$
$\neg\ Coupé \sqcap \neg Convertible \sqcap \neg Hatchback \sqsubseteq Sedan,$
$2Doors \sqcap \exists has0-60mph.\leq_{7.0sec} \sqcap$
$\forall hasKit.SportKit \sqsubseteq SportsCar,$
$Bmw \sqcap Mazda \sqsubseteq \bot,$
$Bmw335i \sqsubseteq Bmw,$
$\exists hasModelYear.\geq_{2012} \equiv NewCar,$
$Bmw \sqsubseteq PrestigiousCar,$
$SportsCar \sqcap Convertible \equiv Roadster,$
$ClassicalKit \sqsubseteq Kit,$
$SportKit \sqsubseteq Kit,$
$Car \sqcap Kit \sqsubseteq \bot,$
$ClassicalKit \sqcap SportKit \sqsubseteq \bot\}$

$\mathcal{A} = \{MazdaMx5Miata(car_1),$
$hasHP(car_1, 167hp),$
$hasFuelConsumption(car_1, 24mpg),$
$hasModelYear(car_1, 2013),$
$has0-60mph(car_1, 6.9sec),$
$hasPrice(car_1, 29960\$),$
$Convertible(car_1),$
$2Doors(car_1),$
$ClassicalKit(kit_1)$
$hasKit(car_1, kit_1),$
$Bmw335i(car_2),$
$hasHP(car_2, 300hp),$
$hasFuelConsumption(car_2, 19mpg),$
$hasModelYear(car_2, 2008),$
$has0-60mph(car_2, 4, 8sec),$
$hasPrice(car_2, 42560\$),$
$Sedan(car_2),$
$4doors(car_2),$
$SportKit(kit_2),$
$hasKit(car_2, kit_2)\}$

$U = \{(InexpensiveCar, 30),$
$(PrestigiousCar, 55),$
$(VeryStrongCar, 50),$
$(StrongCar, 40),$
$(EconomicCar, 30),$
$(NewCar, 35),$
$(Convertible, 10),$
$(Sedan, 5),$
$(\exists hasKit.SportKit, 20),$
$(\exists hasKit.ClasicalKit, 10)\}$

$\mathcal{C} = \{car_1, car_2\}$

Figure 1: The car buyer's background knowledge $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the set of choices $\mathcal{C} = \{car_1, car_2\}$, and preferences encoded as U. We omit the trivial axioms with the super concept Car: $HighClassCar \sqsubseteq Car$, $PrestigiousCar \sqsubseteq Car, \ldots$, etc.

4.1 Implementation

Our Protégé plugin is compatible with both Protégé Desktop version 4.3 and 5.0. As reasoning component we used the Konclude reasoner [24] which turned out to be the best OWL reasoner for our purpose with regards to performance issues.[1] When we start Protégé Konclude is required to be running in the background. The connection to Konclude is established via OWLlink.

Our implementation is straight forward. First, an ontology needs to be loaded via the standard Protégé file menu. This ontology acts as a knowledge base $\mathcal{K}$. After switching to the uDecide tab, the user can specify the set of possible choices by specifying a class C defined in $\mathcal{K}$. All instances of C are treated as choices, which corresponds to the set of choices $\mathcal{C}$ in the theoretical framework. The attributes and their utility can then be specified on top of the vocabulary defined in $\mathcal{K}$. Once the type of choices and the attributes with their corresponding utility have been specified, a connection to Konclude is established via HTTP. We will illustrate these steps in the subsequent section in more details. For each attribute, we request from the reasoner all named individuals satisfying the intersection of the attribute's class expression and the class that defines the type of choices. The result shown consists of a ranked list of all individuals returned by at least one query and their utility which is derived from the satisfied attributes. Since Konclude does not support instance satisfaction queries for anonymous class expressions, we create a temporary ontology that is transferred to Konclude at runtime. We add to this ontology an equivalent classes axiom between each utility assertion's class expression and a named dummy class. We then separately query the individuals for each named dummy class.

As described on our homepage we recommend to configure Konclude to load the knowledge base already on start-up to speed up the calculation. Because of increasing computation time and memory limitations it is required to do this when working with large knowledge bases. If the knowledge base was already loaded into Konclude on start-up only a (very small) temporary ontology needs will be built and used by Konclude. Otherwise the union of both the temporary ontology and the (potentially very big) knowledge base will be loaded in primary memory. This behaviour can be controlled by a checkbox which is used to specify if the knowledge base was already loaded to Konclude at start-up.

4.2 Use Case

As an illustrating use case, we applied our approach to the domain of books and authors. In particular, we used our framework to support a user in finding interesting

[1] We would like to thank Andreas Steigmiller for his support related to using Konclude.

books or authors by specifying his interests as attributes. Instead of working with an artificial example, we used an existing subset of DBpedia that deals with the chosen topic. The core domain contains relevant information about books and their authors. With respect to our use case, DBpedia suffers a bit from its restricted set of terminological axioms and its incompleteness regarding the sparse usage of some properties.

In order to illustrate how to overcome such problems, we decided to extend it with information about cities. In particular, we added for each city the country in which it is located. Furthermore, we added some axioms specifying nationality classes e.g., "Spanish" is defined as "the set of those persons that were born in or died in a city located in Spain or whose nationality is Spain, Spanish_language or Spanish_people". Thus, by using the nationality classes, the nationality of authors for whom no nationality object property assertion exists, can still be inferred by their birth and death places. We took this step because in the core domain the nationality has only been specified directly for 21,3% authors, while 46,9% have a "derived nationality" via our axiomatization. Note that these axioms are not always true, because there obviously exist some people that were born in Spain whose nationality is not Spanish.

This extension illustrates that, in the context of a reasoning based approach, it is possible to leverage background knowledge that seemed not to be relevant at first sight. Information that Barcelona is located in Spain and that some author was born in Barcelona can affect the ranking of choices, if we specified that we prefer Spanish authors as an attribute. It shows also that a reasoning based approach can help to overcome some problems related to incomplete data in the knowledge base. The dataset and some instructions on how to use it can be found at `https://code.google.com/p/udecide/wiki/BookUseCaseExample`.

Suppose that a user wants to find a new author who writes books that are similar to the ones that she likes. First of all, feasible choices have to be defined as the instances of the class *dbp:Author*. Figure 2 depicts a screenshot of the uDecide tab. The class to which the choices belong has been specified in the respective text field in the upper right corner. An arbitrary concept description can be specified as long as it is in the signature of the previously loaded ontology.

Now suppose that our user likes the authors Edgar Stephen King, Allan Poe, and H.P. Lovecraft. Thus, she adds for each of the three authors an attribute to U:

$$U \;=\; \{(\exists influencedBy.\{Stephen\_King\}, 70), \dots\}.$$

The resulting list of attributes can be seen in the uDecide tab on the left side of

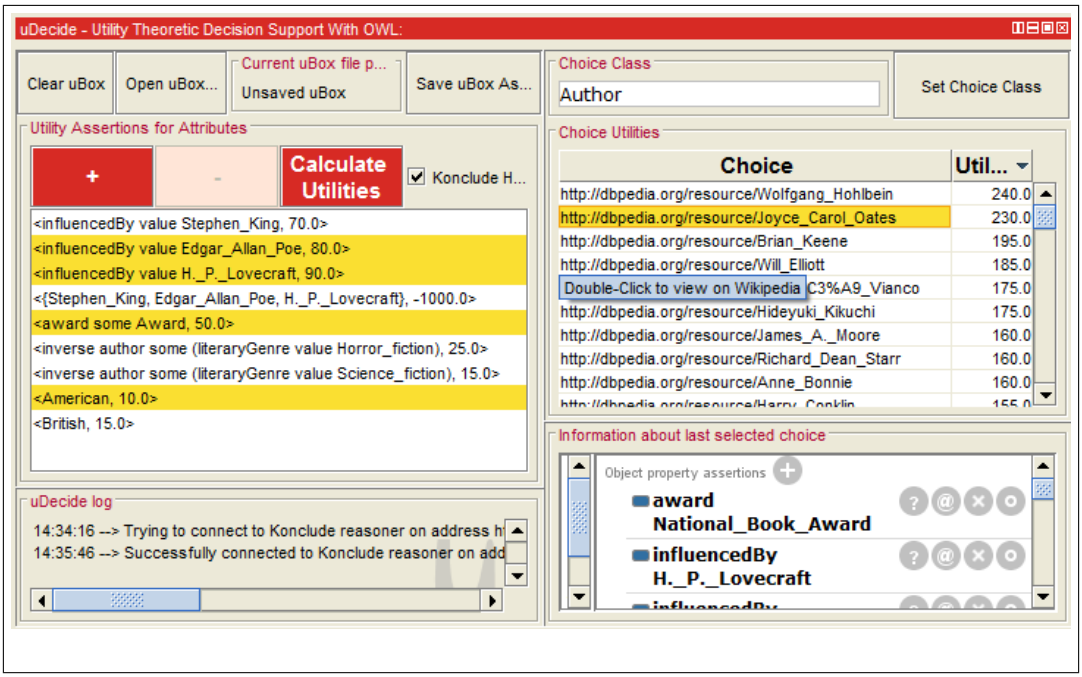

Figure 2: Screenshot of uDecide displaying a ranked list of authors according to the attribute specification of a user.

Figure 2. Note that the concept descriptions are specified in the Manchester syntax[2] supported by the Protégé Editor. All attributes are specified within a dialog box that uses the auto-complete functionality of Protégé as well as its syntax checking capability. Only if a class expression is syntactically correct, a button will be enabled to add it to the UBox.

Overall, nine attributes have been specified. The first three attributes express that the user prefers authors that are influenced by her favorite authors. By adding a negative value to the fourth attribute, the user ensures that the three authors that she already knows will be ranked low in the ranking of choices. The fifth attribute is added to increase the utility of those authors that received some award by 50. Moreover, the user specifies that she likes authors writing books that belong to the genre of horror fiction or science fiction. These attributes have a relatively low utility value. Finally, it is specified that the user likes American and British authors, slightly preferring British.

The results that are finally calculated will only include individuals that satisfy the

[2] `http://www.w3.org/TR/owl2-manchester-syntax/#The_Grammar`

choice class expression and at least one of the attributes. This calculation is started by clicking on the "Calculate Utilities" button. The ranked choices are presented on the right side of Figure 2 in descending order based on their utility. The best choice is the author Wolfgang Hohlbein (240), followed by Joyce Carol Oates (230) and many more lower ranked choices. Thus, the most reasonable choice for the user is to look at the author Wolfgang Hohlbein in more details, given that his attribute specification and the underlying knowledge base is complete and correct.

However, it might often be the case that a user wants to explore the results in more detail, for example to get an explanation about their ranking position. This can be done by clicking on one of the proposed choices. Figure 2 illustrates this for Joyce Carol Oates. The utility score of 230 is based on the fact that Joyce Carol Oates was influenced by Edgar Allen Poe and by H.P. Lovecraft, that she won at least one award and that she was born in New York, therefore being classified as American. Each of the satisfied attributes is highlighted in the left panel. Furthermore, all assertions about the selected choice are shown in a panel in the lower right corner. Again, we have used the Protégé default way of presenting this information. Vice versa, it is also possible to select one of the (or multiple) attributes. This results in those choices being highlighted that satisfy the selected (all the selected) attribute(s) (not shown in Figure 2).

Our use case and the presented example illustrates both the benefits as well as some drawbacks of our approach. First of all, we could apply our Protégé plugin to the domain of books and authors without the plugin requiring any further modifications or extensions. This resulted in an expert system which makes proposals about interesting authors or books. The only required ingredient was an ontology that covers the domain in an appropriate way. We decided to use DBpedia, which features a comprehensive ABox but a flat and inexpressive TBox. Thus, the potential reasoning capabilities of our approach had only a limited impact: whether a choice satisfies an attribute can be decided by a direct look-up over most attributes. This changes with the use of an expressive TBox where some of the attributes are satisfied due to a chain of non-trivial logical dependencies and background knowledge. In such a setting our approach can be used to provide non-immediately obvious recommendation, as well as elicit their *explanation*, a key issue in decision support.

4.3 Scalability

In some preliminary experiments we have also tested the scalability of our approach. We used both (1) standard datasets of our DBpedia book use case as well as (2) extended datasets (in the sense that new –not present in DBpedia– terminological and assertional knowledge was added). For each case we created subsets that differed

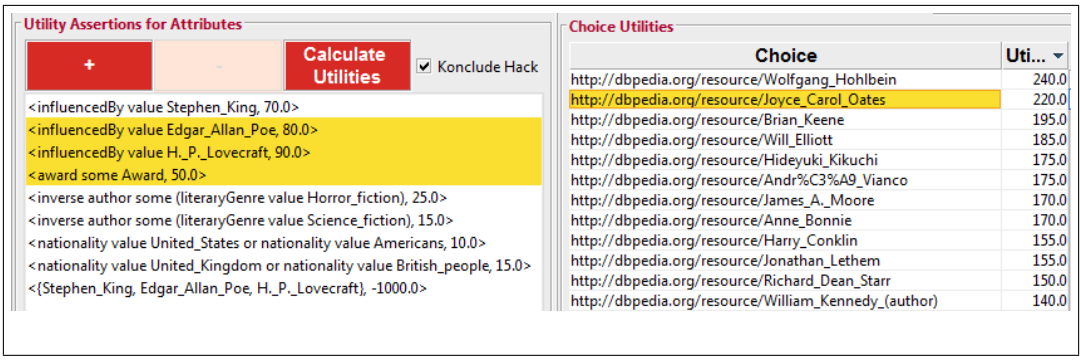

Figure 3: Screenshot of uDecide performance test displaying a ranked list of authors using the *standard* DBpedia book knowledge base.

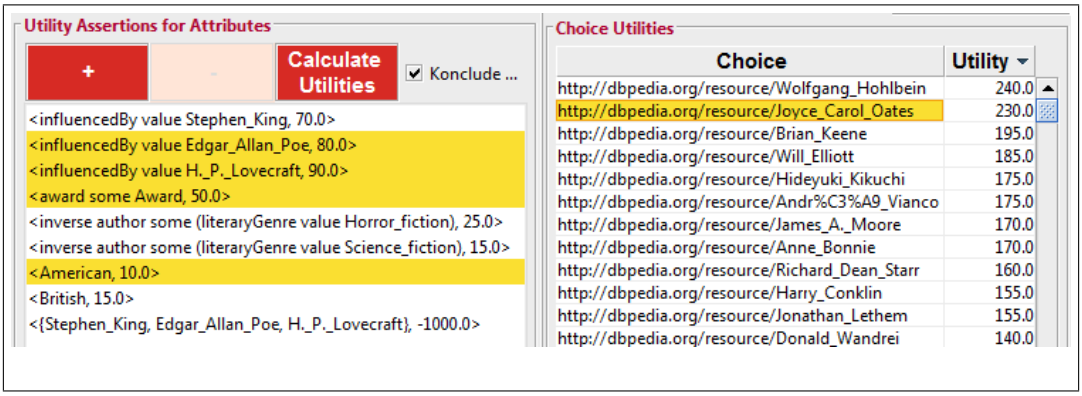

Figure 4: Screenshot of uDecide performance test displaying a ranked list of authors using the *extended* DBpedia book knowledge base.

with respect to the contained number of instances. The smallest dataset contained 13307 individuals and 51149 assertions and axioms. The largest dataset contained 54018 individuals and 300653 assertions and axioms covering all authors and books in DBpedia. The extended datasets are slightly larger in each case because they contain assertions about the countries of the cities. To further highlight the differences between the standard and the extended datasets, we used different UBoxes to measure their runtime. The UBoxes as well as the resulting choice ranking for the complete dataset can be seen in Figures 3 and 4.

The UBoxes share most of their attributes, differing only on the attributes giving preference to authors with American and British nationality. In the standard case they were expressed only with the nationality object property on the domain of *dbp:Author* while in the extended case the added nationality classes were used. The

Table 1: Runtime results for different sizes of the DBpedia standard book knowledge base.

Size	# Individuals	# Axioms	Runtime
66 MB	53991	297926	5.8 sec
55 MB	47311	249126	4.6 sec
44 MB	40044	199898	4.1 sec
33 MB	33253	150900	3.1 sec
22 MB	23456	101162	2.1 sec
11 MB	13307	51149	1.0 sec

Table 2: Runtime results for different sizes of the DBpedia extended book knowledge base.

Size	# Individuals	# Axioms	Runtime
66 MB	54018	300653	8.2 sec
55 MB	47338	251454	6.9 sec
44 MB	40068	201883	5.3 sec
33 MB	32114	152023	3.8 sec
22 MB	23487	102357	2.6 sec
11 MB	13335	51893	1.3 sec

results of our experiments are shown in Tables 1 and 2.

We conducted our experiments on a Win 7 desktop machine with four 3,4 Ghz cores and 8 GB DDR3-Ram using Protege 5.0 Desktop Beta build 17 and Konclude-v0.6.1-527-Windows-x64-VS05-Dynamic-Qt4.8.5. As described above, we forced Konclude to load the knowledge base already on start-up. This required less than 4 seconds for each of the datasets listed in the table. It can be seen that the runtime behavior of uDecide is linear with respect to the size of the knowledge base. However, the runtime of uDecide is mainly based on the reasoning system that is used in the background which was Konclude in our experiments. It can also be observed that the extended case takes more time than the standard case. This is no surpise, given that the added nationality classes, being rather complex DL concepts, required more expressive reasoning than the original “flat" attributes. In any case, even for the complete and extended dataset, we measured a runtime of not more than 8.2 seconds, which is probably still acceptable for an application where a user is waiting for an on-the-fly response.

As already described above, our experiments are currently based on an the DBpedia ontology. Further experiments have to be conducted where we use more expressive knowledge bases. We are not expecting a linear runtime behavior in such a setting.

5 Related Work

On the theoretical side, many approaches for preference representations are based on propositional logic [6, 11, 17, 26] and also modal logic. DL languages have also been used for preference representation by [22, 21]. In their work, the utility of a concept (proposal) is defined as the sum of the weights of its superconcepts. We should note that their preference set, which is a set of weighted concepts, is similar

to our UBox. However, differently from us, they represent choices also as concepts. Moreover, we provide an efficient implementation of our framework. In [25], authors show how to encode fuzzy MCDM problems in the formalism of fuzzy DL. They base their work on a standard feature of MCDM on continuos domains: a *decision matrix* wherein the performance score of each choice over each criteria is explicitly stated. Criteria are expressed as fuzzy concepts. The optimal choice (w.r.t the fuzzy knowledge base) is the one with the highest *maximum satisfiability degree*. The authors do not explicitly make a distinction between the knowledge base and the set of criteria. In general, the focus of the work is to show the potential and flexibility of fuzzy DL in encompassing the usual numerical methods used in MCDM, rather than leveraging a formal concept hierarchy in MCDM for expressing relations and handling inconsistencies between criteria, choices, and the knowledge base.

On the practical side, to our knowledge there are two decision support system that are built on top of Protégé , that is of relevance. One of them is [8] and the other is [10]. It is relevant to note that those systems are designed mainly as a clinical decision support system, therefore they use clinical guidelines. Our plugin is established on a utility-theoretic basis, and aimed to be developed as a rather general decision making framework.

6 Conclusion and Further Work

We have presented a theoretical framework for describing multiattribute decision problems in terms of weighted DLs. A preliminary version of this framework was first described in [2]. Within this paper we have improved this framework and presented a refined version. However, the main contribution of this paper is the Protégé plugin uDecide. This plugin is a straight forward implementation of the proposed framework. It computes the utility for each of the defined choices by aggregating the utility value for each satisfied attribute. Since each attribute corresponds to a class description, standard reasoning techniques can be used to check whether an attribute is satisfied. We have used the Konclude [24] reasoning system to conduct the required reasoning tasks. The results of this computation are presented to the user as a ranked list of choices.

To our knowledge we have introduced the first system that allows to encode and solve multiattribute decision problems in terms of weighted DLs. Since we implemented our approach as a Protégé plugin, our approach can easily be used by the DL community. The current form of uDecide works with the u_σ utility function and we are currently working on extending the plug-in as an implementation of a generic decision base, where one can also define arbitrary utility functions, along

with choice criteria (choice function).

We have demonstrated within our use case, where we used a DBpedia fragment concerned with the book domain, how to use uDecide as an expert system that recommends new authors to a user. Moreover, we have also shown that our current implementation, by using the reasoning system Konclude, is capable to deal with large real-world datasets. We already pointed out that the benefits of reasoning are rather limited in the context of the DBpedia subset we used. For that reason, we have to identify another use case where we can clearly show that reasoning is beneficial by making logical dependencies explicit in calculating the final ranking. It will also be interesting to perform runtime experiments on the dataset of such a use case. We are currently investigating datasets from the biomedical domain and from the domain of life sciences.

As in many other disciplines, in decision theory it is common to deal with decisions where uncertainty is present. For that reason, one major future research direction is to extend the framework with probabilistic description logics, e.g., [18, 19]. A first attempt in that direction can be found in [1]. A probabilistic approach would allow us to face the challenge for representing typical problems defined in the decision theoretic literature, along with lots of new application possibilities. In particular, the probabilistic extension would allow us to compute the expected utility of choices (lotteries) in terms of their logical implications along with the type of the probability that the theory conforms (e.g., subjective, statistical). A second major research direction is to extend the framework to sequential decisions (e.g. $\mathcal{D}_i \rightarrow \mathcal{D}_{i+1}$, sequence of decision bases). Once sequential decisions are defined, we can deal with policies, strategies and also planning. Moreover, we believe that, with a proper theoretical extension, uDecide could be used to reason in multi agent environments and applied to computational social choice or algorithmic game theory.

References

[1] E. Acar. Computing subjective expected utility using probabilistic description logics. In U. Endriss and J. Leite, editors, *STAIRS*, volume 264 of *Frontiers in Artificial Intelligence and Applications*, pages 21–30. IOS Press, 2014.

[2] E. Acar and C. Meilicke. Multi-attribute decision making using weighted description logics. In T. Lukasiewicz, R. Peñaloza, and A.-Y. Turhan, editors, *PRUV*, volume 1205 of *CEUR Workshop Proceedings*, pages 1–14. CEUR-WS.org, 2014.

[3] F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In *IJCAI*, pages 364–369. Professional Book Center, 2005.

[4] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.*

Cambridge University Press, 2003.

[5] C. Boutilier. Toward a logic for qualitative decision theory. In J. Doyle, E. Sandewall, and P. Torasso, editors, *kr94*. MK, SF, 1994.

[6] S. Bouveret, M. Lemaître, H. Fargier, and J. Lang. Allocation of indivisible goods: a general model and some complexity results. In *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, pages 1309–1310. ACM, 2005.

[7] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[8] M. Ceccarelli, A. Donatiello, and D. Vitale. KON^3: A clinical decision support system, in oncology environment, based on knowledge management. In *ICTAI (2)*, pages 206–210. IEEE Computer Society, 2008.

[9] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 363–369, Menlo Park, CA, July 30– 3 2000. AAAI Press.

[10] C. Chen, K. Chen, C.-Y. Hsu, and Y.-C. J. Li. Developing guideline-based decision support systems using protégé and jess. *Computer Methods and Programs in Biomedicine*, 102(3):288–294, 2011.

[11] Y. Chevaleyre, U. Endriss, and J. Lang. Expressive power of weighted propositional formulas for cardinal preference modelling, Dec. 08 2006.

[12] P. C. Fishburn. *Utility Theory for Decision Making*. Robert E. Krieger Publishing Co., Huntington, New York, 1969.

[13] P. H. Giang and P. P. Shenoy. Two axiomatic approaches to decision making using possibility theory. *European Journal of Operational Research*, 162(2):450–467, Apr. 16 2005.

[14] S. Kaci and L. van der Torre. Reasoning with various kinds of preferences: logic, non-monotonicity, and algorithms. *Annals OR*, 163(1):89–114, 2008.

[15] C. Kahraman. *Multi-Criteria Decision Making: Theory and Applications with Recent Developments*. Springer, 2008.

[16] R. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[17] C. Lafage and J. Lang. Logical representation of preferences for group decision making. In *KR2000: Principles of Knowledge Representation and Reasoning*, pages 457–468, San Francisco, 2000. Morgan Kaufmann.

[18] T. Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6–7):852–883, 2008.

[19] C. Lutz and L. Schröder. Probabilistic description logics for subjective uncertainty. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010*. AAAI Press, 2010.

[20] G. Parmigiani and L. Y. T. Inoue. *Decision Theory Principles and Approaches*. John Wiley & Sons, Ltd, 2009.

[21] A. Ragone, T. D. Noia, F. M. Donini, E. D. Sciascio, and M. P. Wellman. Weighted description logics preference formulas for multiattribute negotiation. In *Proceedings of Scalable Uncertainty Management, Third International Conference, SUM 2009*.

[22] A. Ragone, T. D. Noia, F. M. Donini, E. D. Sciascio, and M. P. Wellman. Computing utility from weighted description logic preference formulas. In *DALT*, volume 5948 of *Lecture Notes in Computer Science*, pages 158–173. Springer, 2009.

[23] Y. Shoham. Conditional utility, utility independence, and utility networks. *CoRR*, abs/1302.1568, 2013.

[24] A. Steigmiller, T. Liebig, and B. Glimm. Konclude: system description. *Web Semantics: Science, Services and Agents on the World Wide Web*, 27:78–85, 2014.

[25] U. Straccia. Multi criteria decision making in fuzzy description logics: A first step. In *Knowledge-Based and Intelligent Information and Engineering Systems, 13th International Conference, KES 2009*.

[26] D. Zhang and Y. Zhang. A computational model of logic-based negotiation. In *AAAI*, pages 728–733. AAAI Press, 2006.

Received 26 July 2015